Terraform Task

Name: Shaik Khaja Basha
Batch : Batch 11
Date : 26.07.2025
Task : Azure CLI

Azure CLI

The Azure Command-Line Interface (CLI) is Microsoft's cross-platform command-line tool for managing Azure resources. It allows you to create, manage, and monitor Azure resources from the command line or in scripts.

- Its helps to interact with Azure services to manage resources

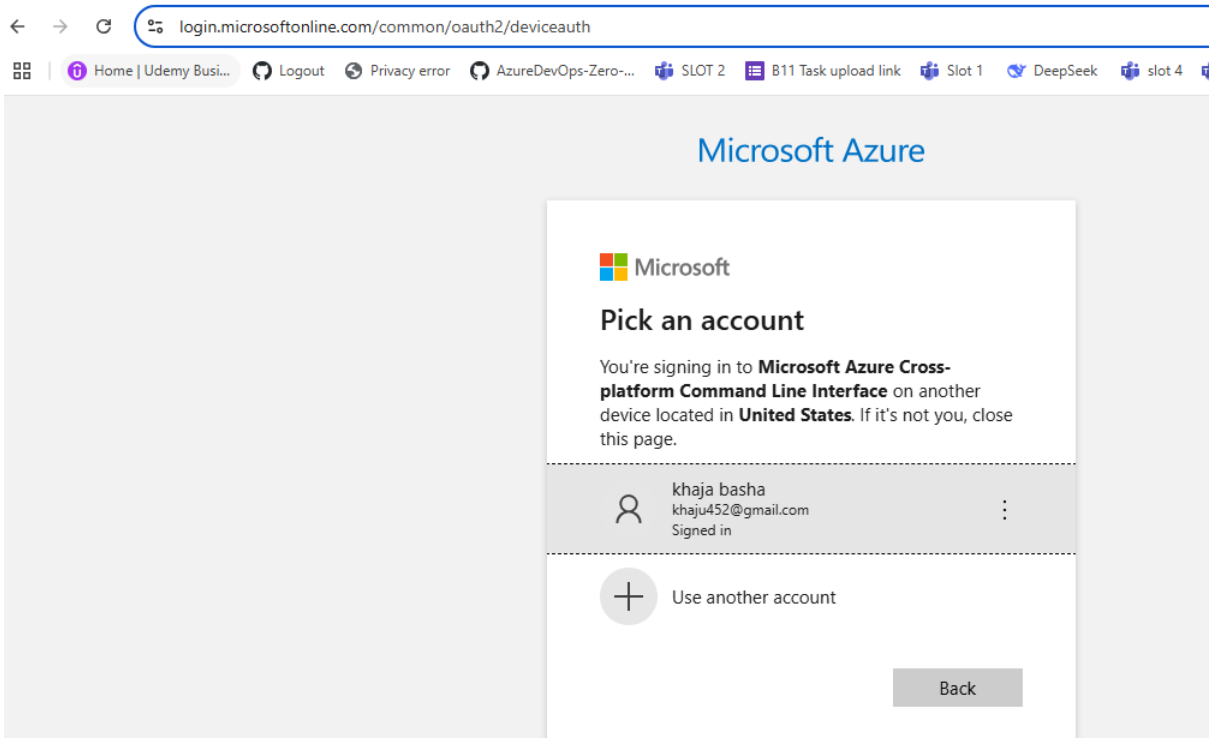
Basic Commands

Login

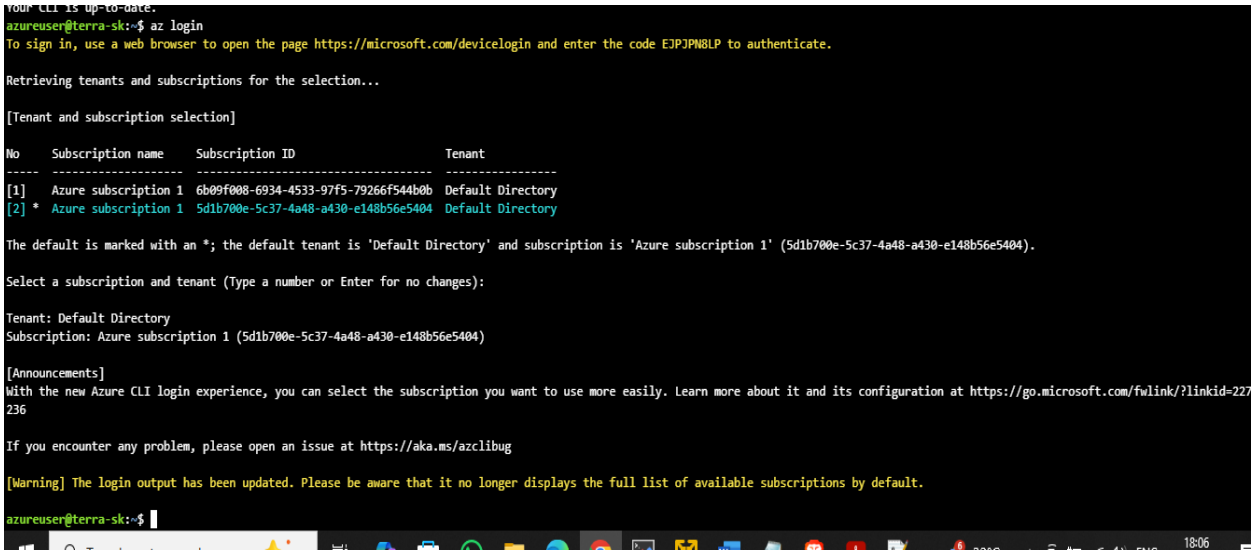
az login

```
Your CLI is up-to-date.
azureuser@terra-sk:~$ az login
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code EJPJPN8LP to authenticate.
```

- When we write az login it will create a link with code
- Copy the code and click the URL it will open a window and login with credentials



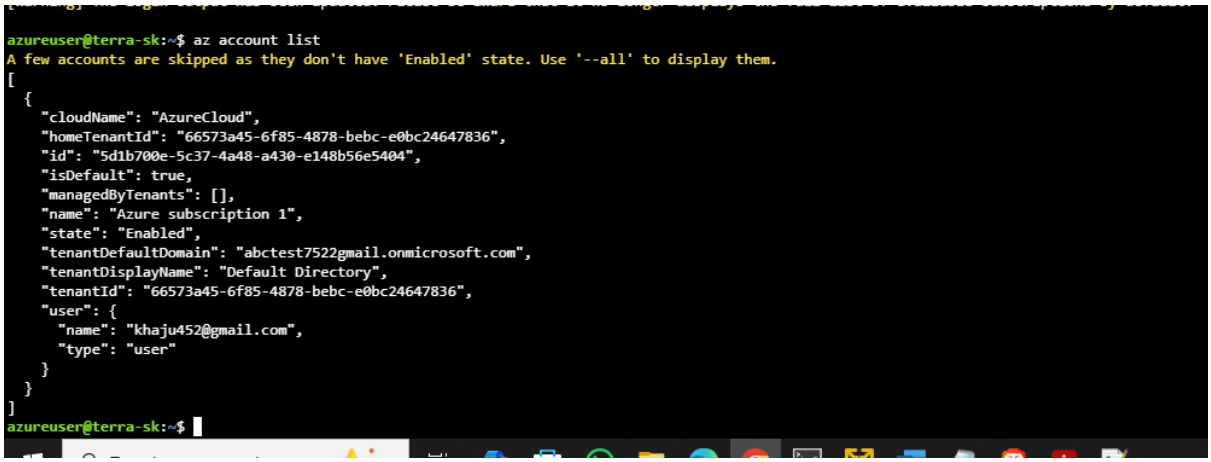
As soon as login it will give me the tenants which will be there



List subscriptions

az account list

which will show what are the account subscription

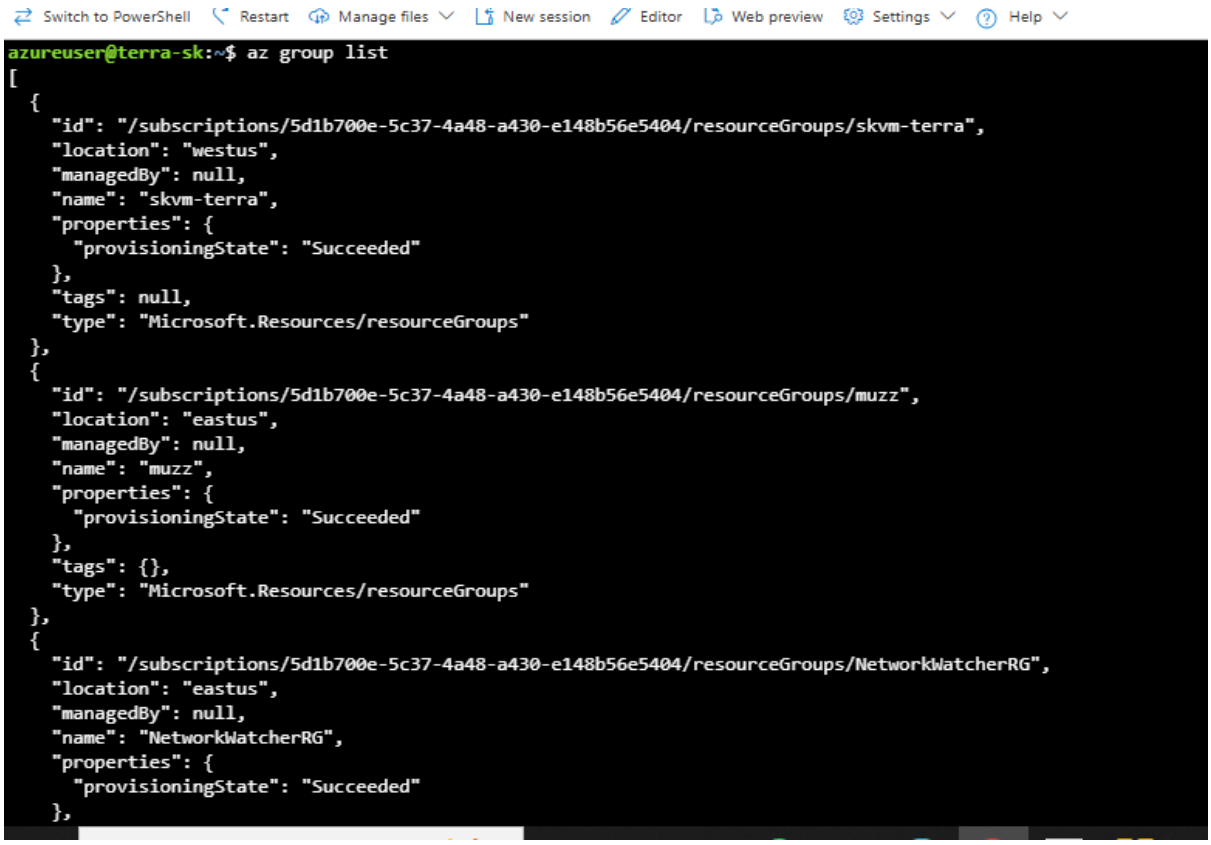


Set default subscription

az account set --subscription "My Subscription Name"

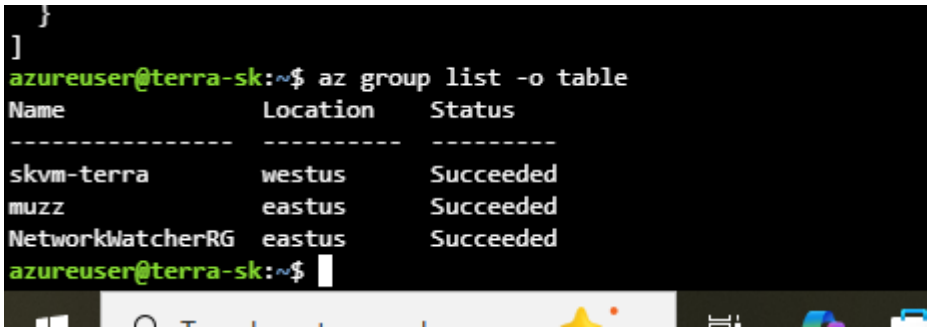
List resource groups

az group list



Which will give me as a JSON format if I want to see as table then

Az group list -o table

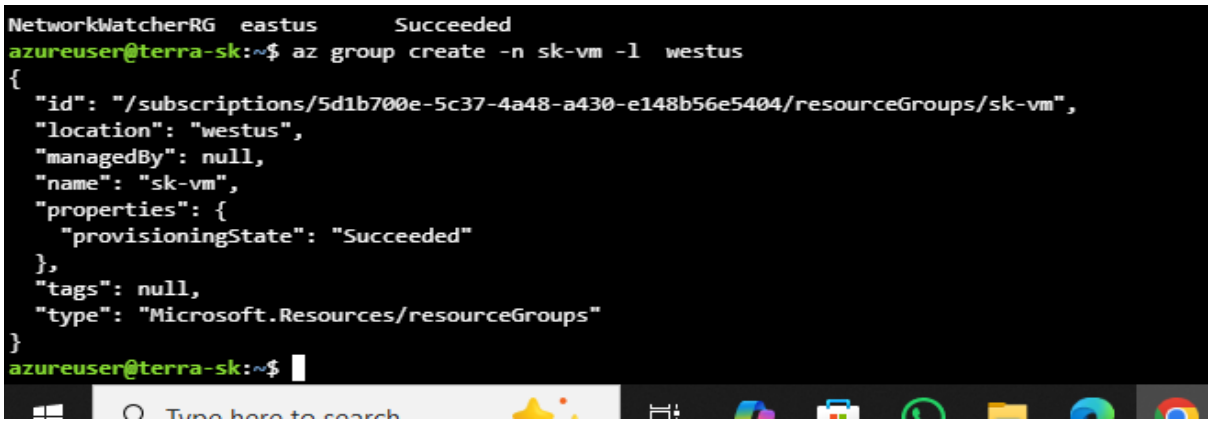


Which is easy for understanding

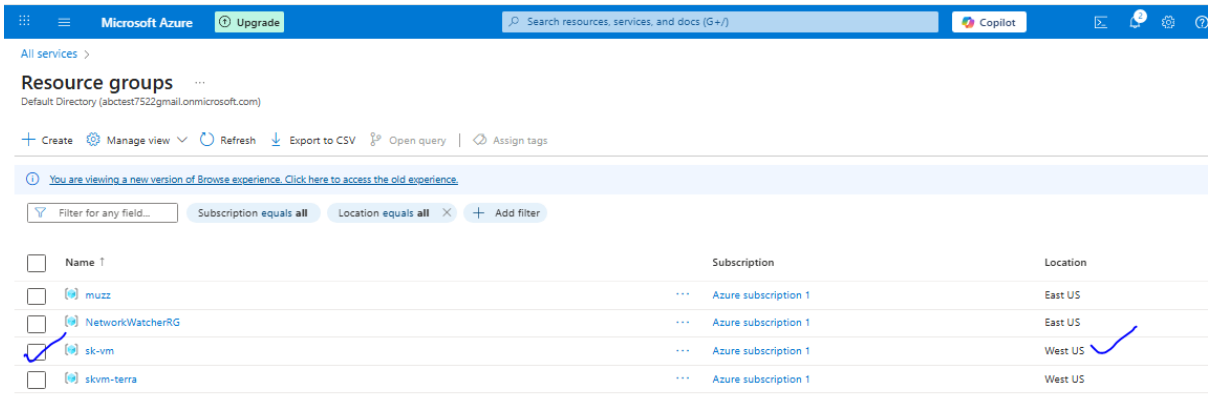
Create a resource group

az group create --name MyResourceGroup --location eastus

az group create -n sk-vm -l westus



We can check whether its created or not though GUI



List virtual machines

az vm list

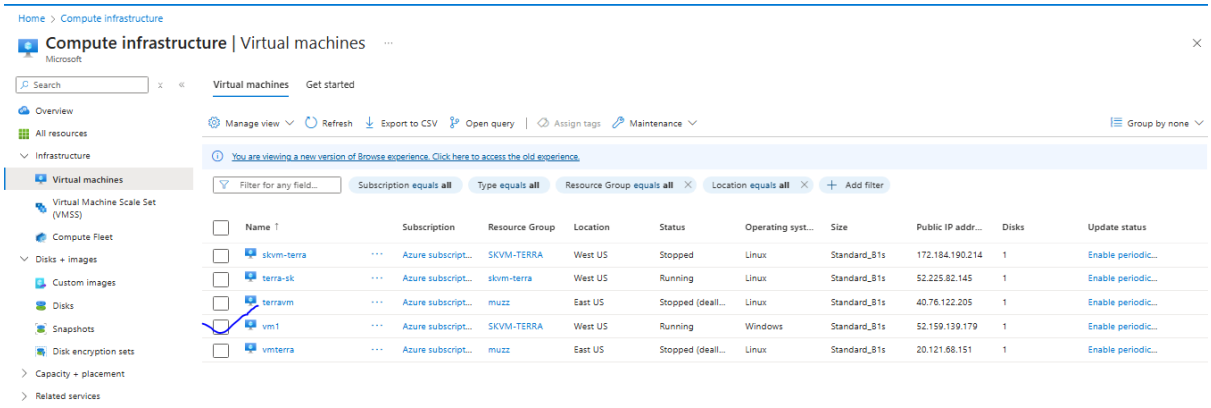
We can start or stop the vm through cli

az vm start -g <rgname> -n <vmname>

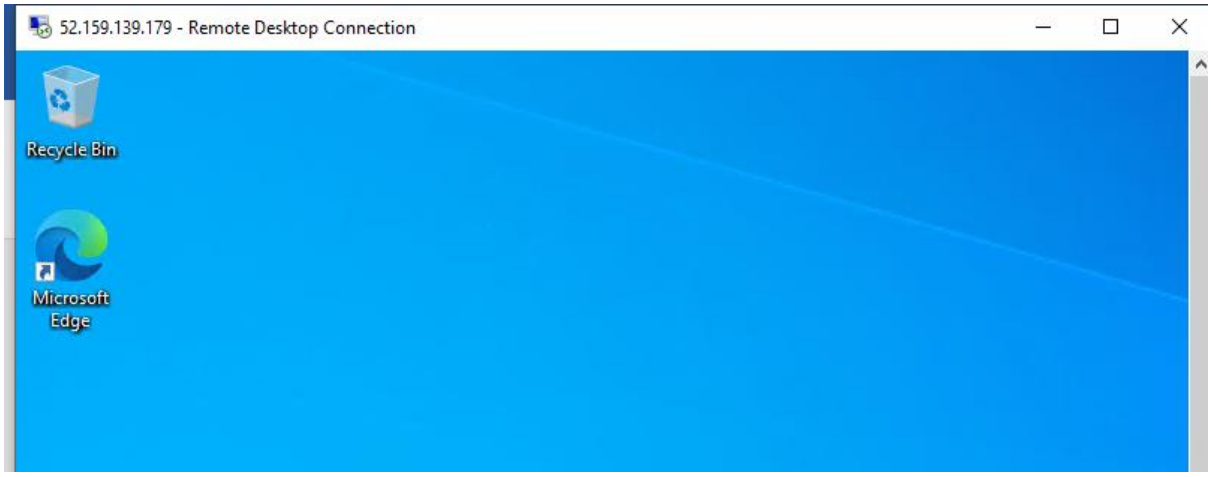
az vm show -g <rgname> -n <vmname>

creating a vm

az vm create -g test -n vm1



We can see a vm is created by using azure cli



Create azure active directory(AAD) or entra id

Checking Existing Directories

az ad signed-in-user show --query 'userPrincipalName'

az account list --query '[].tenantId' -o tsv

lets connect azure to terraform

az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/5d1b700e-5c37-4a48-a430-e148b56e5404"

if its get fail we can use export command with required details

{

"appId": " 1ba6dbc4-1815-4d1d-b576-ef9060254d0a ",

"displayName": "5d1b700e-5c37-4a48-a430-e148b56e5404",

"password": " BsT8Q~knW3dH2~C5wlZDf8aibZdIel7xEUocSbYN ",

```
  "tenant": "66573a45-6f85-4878-bebc-e0bc24647836"
}
```

```
subscription_id = " 5d1b700e-5c37-4a48-a430-e148b56e5404"
  client_id      = " 1ba6dbc4-1815-4d1d-b576-ef9060254d0a"
  client_secret   = " BsT8Q~knW3dH2~C5wlZDf8aibZdIel7xEUocSbYN"
  tenant_id      = " 66573a45-6f85-4878-bebc-e0bc24647836"
```

```
export ARM_CLIENT_ID="1ba6dbc4-1815-4d1d-b576-ef9060254d0a "
export ARM_CLIENT_SECRET=" BsT8Q~knW3dH2~C5wlZDf8aibZdIel7xEUocSbYN"
export ARM_TENANT_ID="66573a45-6f85-4878-bebc-e0bc24647836"
export ARM_SUBSCRIPTION_ID="5d1b700e-5c37-4a48-a430-e148b56e5404"
```

```
khaja [ ~ ]$ az account set --subscription "5d1b700e-5c37-4a48-a430-e148b56e5404"
khaja [ ~ ]$ export ARM_CLIENT_ID="1ba6dbc4-1815-4d1d-b576-ef9060254d0a "
export ARM_CLIENT_SECRET=" BsT8Q~knW3dH2~C5wlZDf8aibZdIel7xEUocSbYN"
export ARM_TENANT_ID="66573a45-6f85-4878-bebc-e0bc24647836"
export ARM_SUBSCRIPTION_ID="5d1b700e-5c37-4a48-a430-e148b56e5404"
khaja [ ~ ]$ mkdir azure
```

Create a directory mkdir azure

Create a resource provider file vi res.tf

```
terraform {
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = "4.37.0"
    }
  }
}
provider "azurerm" {
  features {}
}
```

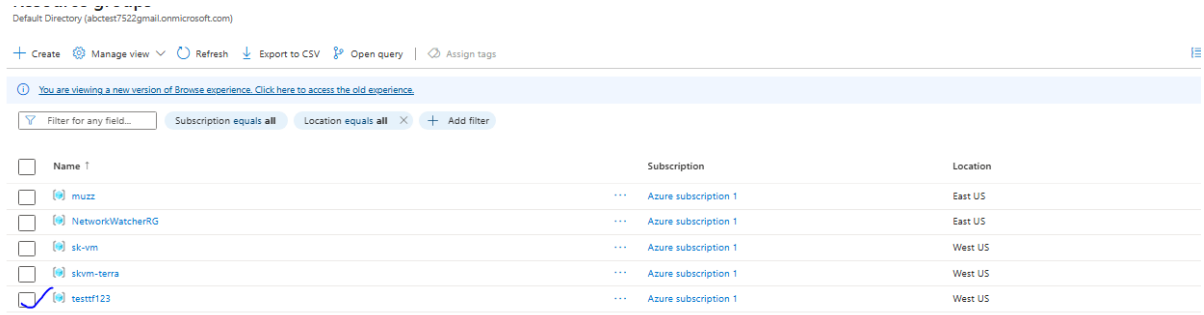
Then we initialized the terraform init

```
export ARM_SUBSCRIPTION_ID="5d1b700e-5c37-4a48-a430-e148b56e5404"
khaja [ ~ ]$ mkdir azure
khaja [ ~ ]$ cd azure/
khaja [ ~/azure ]$ vi res.tf
khaja [ ~/azure ]$ vi res.tf
khaja [ ~/azure ]$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/azurerm versions matching "4.37.0"...
- Installing hashicorp/azurerm v4.37.0...
- Installed hashicorp/azurerm v4.37.0 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
khaja [ ~/azure ]$
```

```
Message: The subscription '66573a45-6f85-4878-bebc-e0bc24647836' could not be found.
khaja [ ~/azure ]$ az ad sp create-for-rbac --role="Contributor" --scopes="/subscriptions/5d1b700e-5c37-4a48-a430-e148b56e5404"
Creating 'Contributor' role assignment under scope '/subscriptions/5d1b700e-5c37-4a48-a430-e148b56e5404'
The output includes credentials that you must protect. Be sure that you do not include these credentials in your code or check the credentials into your source control. For more information,
see https://aka.ms/azadsp-cli
{
  "appId": "94a1cec8-3d1b-4114-b742-1b400c2f2252",
  "displayName": "azure-cli-2025-07-27-14-59-48",
  "password": "wq18Q~NDo4B_fFSA.oivpmrcBbsy63uaTumUjb8S",
  "tenant": "66573a45-6f85-4878-bebc-e0bc24647836"
}
khaja [ ~/azure ]$ az account list -o table
Name                 CloudName    SubscriptionId                        TenantId                              State    IsDefault
-------------------  -----------  ------------------------------------  ------------------------------------  -------  -----------
Azure subscription 1  AzureCloud  5d1b700e-5c37-4a48-a430-e148b56e5404  66573a45-6f85-4878-bebc-e0bc24647836  Enabled  True
khaja [ ~/azure ]$ export ARM_CLIENT_SECRET="wq18Q~NDo4B_fFSA.oivpmrcBbsy63uaTumUjb8S"
khaja [ ~/azure ]$ export ARM_CLIENT_SECRET=" wq18Q~NDo4B_fFSA.oivpmrcBbsy63uaTumUjb8S"
khaja [ ~/azure ]$ export ARM_CLIENT_ID="94a1cec8-3d1b-4114-b742-1b400c2f2252 "
khaja [ ~/azure ]$ terraform apply

No changes. Your infrastructure matches the configuration.
```

When we do terraform apply the resources are created

```
khaja [ ~/azure ]$ terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # azurerm_resource_group.rg will be created
  + resource "azurerm_resource_group" "rg" {
      + id       = (known after apply)
      + location = "westus"
      + name     = "testtf123"
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

azurerm_resource_group.rg: Creating...
azurerm_resource_group.rg: Still creating... [00m10s elapsed]
azurerm_resource_group.rg: Creation complete after 17s [id=/subscriptions/5d1b700e-5c37-4a48-a430-e148b56e5404/resourceGroups/testtf123]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
khaja [ ~/azure ]$
```



Here we can see the resource has been created

If I don't want to give export everytime we can provide in our configuration once

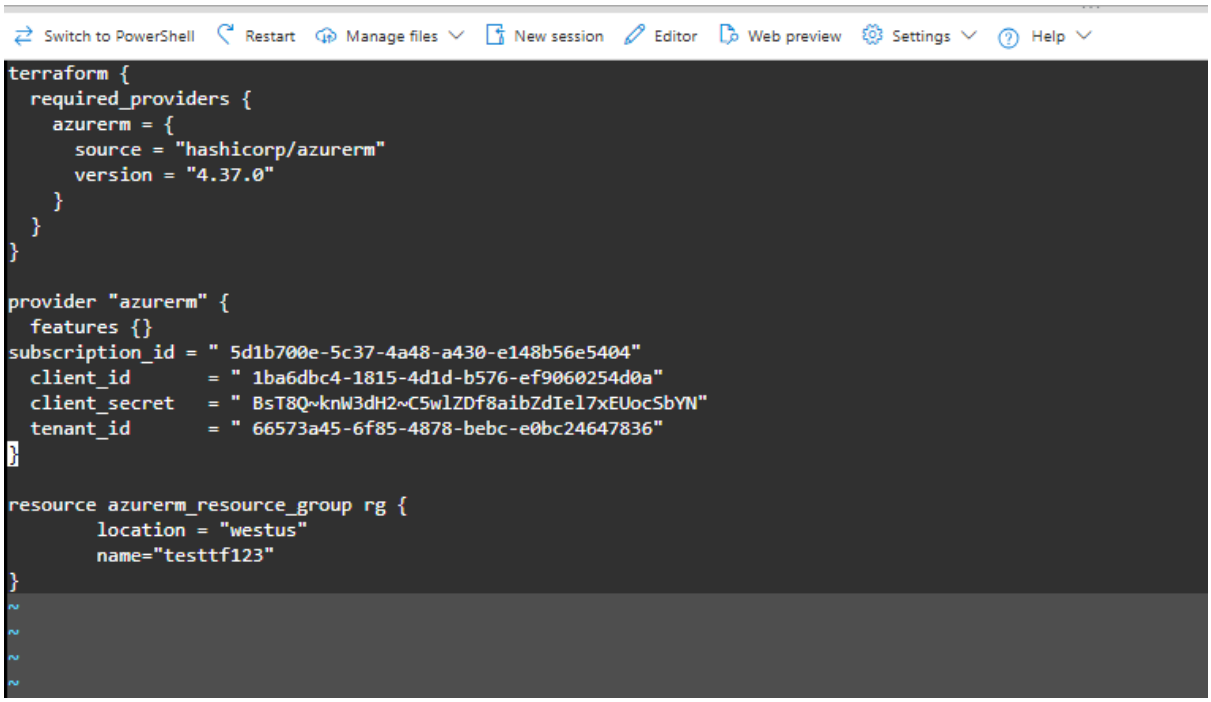
subscription_id = "bccafdc0-1217-4202-883e-2c0dbce2e3cb"

client_id = "90638e0f-524e-45ea-b751-7f200d04a9c9"

client_secret = "35M8Q~sGvaujHKbwi0jDgLxkMvj7VX0PqnSYgabq"

tenant_id = "c9947813-2c41-49f2-937e-98e030d55bae"

}



Which is not preferable always use environmental variables